Usage Inquiry System User's Guide

I. P. Sharp Associates

SHARP APL/370
Release 19

APL Systems Development Group
I. P. Sharp Associates Limited
2 First Canadian Place, Suite 1900
Toronto, Ontario, Canada
M5X 1E3

Telephone 416 364-5361
Telex 0622259 I P SHARP TOR

PREFACE

This document describes how the use of APL system resources can be reported with functions in workspace 1 *USAGE*.

This user guide replaces SATN 9 "Usage Inquiry System", code: 0132 8303 E4.

Printed in Canada
March 1987
Publication Code: 0132 8703 E5

CONTENTS

INTRODUCTION, 1
CODES, 2
MASKS, 2
ACCESS TO USAGE DATA, 3
FUNCTIONS IN WORKSPACE 1 *USAGE*, 3
- *ACCESS*, 3
- *ACCOUNTS*, 3
- *BCCC*, 4
- *BFDEFN*, 4
- *BF1DAYDEFN*, 4
- *CUSTADDR*, 4
- *GROUPS*, 4
- *NAMES*, 5
- *OFFHIST*, 5
- *OHDEFN*, 6
- *OHONLINE*, 6
- *OHPRINT*, 6
- *ONLINE*, 7
- *PRINT*, 7
- *QUOTAS*, 8
- *STORAGE*, 8
- *UPDATES*, 9
- *USAGE*, 9
- *WHOIS*, 9

APPENDIX A: FUNCTION DEFINITIONS, 11
INDEX, 13

INTRODUCTION

You use the functions in workspace 1 *USAGE* to access information about the usage of system resources. All users may access data relating to their own usage. Where appropriate access has been obtained, you may use the same functions to access data relating to usage by other user numbers.

All the data collection functions are self-contained, and can be copied into a user's workspace for the construction of any form of reporting or monitoring package.

Access to usage data for other user numbers is controlled by Codes and Masks. For the user who is interested only in accessing his own usage data, Codes and Masks are unnecessary.

CODES

Every user number in the APL system is made up of two integer numbers: a branch code and a customer code; each made up of several parts. Together they form a code, which uniquely defines an invoicing group of user numbers for billing purposes, and appears on the monthly APL invoice.

The branch code contains information related to I.P. Sharp Associates, and is made up of three parts. The first part is the I.P. Sharp company number, which is one or more digits. Two digits are added to the right of the company number to define the I.P. Sharp office within the company. A further two digits are added to the right of both the company and the office to define the department within the I.P. Sharp office.

The customer code consists of customer-related information, and has two parts. The first part is the customer number, usually of three or more digits. Three digits are appended to the right of the customer number to define an invoice subgroup for that customer number. A subgroup number is meaningless without a customer number to the left of it. For example, the code:

```
14231 4321001
```

uniquely defines one billing group to which user numbers may be assigned. Inspection of the branch code, 14231, reveals that the invoice group is within department 31 of office 42 of company 1. Inspection of the customer code reveals that this is invoice group 001 for customer number 4321.

Note that to maintain the correct number of digits, fields are zero filled. However, none of the parts of a code (company, office, department, customer, or invoice numbers) may have a zero value.

MASKS

A mask is a means of indicating to the *USAGE* functions a range of codes. If the rightmost fields in either the branch code or customer code are set to zero, any value is acceptable in these fields.

A branch code mask of:

14231	indicates department 31 of office 42 of company 1
14200	indicates any department in office 42 of company 1
10000	indicates any department in any office of company 1
00000	indicates any possible company, office, and department combination

A customer code mask of:

4321001 indicates invoice group 001 for customer 4321

4321000 indicates all invoice groups for customer number 4321

0000000 indicates all possible customer and invoice group combinations

Of course 00000 and 0000000 are identical to 0, and so 0 may be used as the most general mask for either the branch code or the customer code masks.

A branch code mask and a customer code mask may be combined to create a mask. For example, to indicate that all codes in company 3 for customer 1234 are acceptable, the mask would be:

```
30000 1234000
```

To indicate all codes for customer number 1234 the mask would be:

```
0 1234000
```

ACCESS TO USAGE DATA

All users have access to their own usage data. Access to more than your own usage data can be arranged through your local SHARP APL representative.

Access is controlled by the use of mask. If a user has no access mask, he has default access to his own usage data. Otherwise, each user has his own set of access masks, each additional access mask denoting a range of codes to which he has access. Usage data is available to the user through the use of the functions in workspace 1 *USAGE*.

FUNCTIONS IN WORKSPACE 1 *USAGE*

R←*ACCESS*

The result is an *N*×2 matrix of the user's access control masks. If the result is an empty matrix, then the user has access only to his own data.

R←*ACCOUNTS* **mask**

The right argument **mask** is one or more masks. **mask** may be any shape provided that the ravel of mask results in pairs. If the argument is empty, or does not have an even number of elements, it is zero-filled to be one or more pairs in length.

The result is an *N*×3 numeric matrix, one row for each user number available to the accessor, whose codes fall within the mask. The first column is the user number and the second and third columns the corresponding codes.

R←*BCCC* **user numbers**

The argument **user numbers** is a scalar or vector containing the user numbers for which the codes are required.

The result is an *N*×3 numeric matrix, one row for each user number in the argument. Column 1 is the user number, and columns 2 and 3 the corresponding codes.

If any of the numbers in the argument are not in the APL system, or are unavailable to the accessor, then the corresponding row is zero-filled.

BFDEFN

Defines several variables which may be used to index the result of the function *USAGE* (see Appendix A).

BF1DAYDEFN

Defines several variables which may be used to index the result of the function *STORAGE* (see Appendix A).

R←*CUSTADDR* **code**

The argument **code** is a vector or matrix of the code(s) for which the customer address is required. More than one code may be specified as an *N*×2 matrix.

The result is an 8×50 character matrix of the customer address, or an *N*×8×50 character array if the argument is a matrix.

If the code does not exist, or is unavailable to the accessor, then the matrix is blank-filled.

R←*GROUPS* **code**

The argument **code** is a vector or a matrix. If the argument is a vector, it consists of one or more codes; if it is a matrix, each row may contain one or more codes.

The result is an *N*×2 numeric matrix with each row representing a unique code. *GROUPS* returns all codes within a given branch code or customer code.

The following are valid uses of *GROUPS*:

GROUPS 10101 4356000
Return all branch codes and customer codes for customer 4356 in location 10101

GROUPS 10101 4356000 10111 547000
Return all branch codes and customer codes for 4356 in 10101 and 547 in 10111

GROUPS 10000 4356000 20000 4356000
Return all branch codes and customer codes for 4356 in countries 1 and 2

GROUPS 3 2ρ0 4356000 0 543000 10101 0
Return all branch codes and customer codes for customers 4356 and 543 in all countries and for all customers in branch 10101

R←NAMES **user numbers**

The argument user numbers is a scalar or vector, containing the user numbers for which the signon names are required.

The result is an *N*×22 character matrix of names, one row for each user number in the argument. The format of the result is similar to the format of □*NAMES* or □*LIB*.

If any of the numbers in the argument are not in the APL system, or are unavailable to the accessor, then the corresponding row is blank-filled.

R←OFFHIST **myda**

The argument **myda** is a vector of:

monthnumber year [daynumbers] [usernumbers]

If no days are specified, all days are reported. If no user numbers are specified, the current user's number is assumed.

R is a matrix of signoff records for all user numbers that the user requested and may access. The function *OHDEFN* defines variables which give meaning to the values in each column.

For example:

OFFHIST 8 79
Return all records for the accessor for August 1979

OFFHIST 6 1979 25 26 27
Return all records for 25, 26, and 27 June 1979

OFFHIST 10 78 2381946 4511127
Return all specified users for October 1978

OFFHIST 8 79 1 2 4511127 3918261
Return all users on 1 and 2 August 1979

R←file *OFFHIST* myda

The left argument *FILE* may be the number of a file which is already tied, or a file name which will be created if possible, and if it doesn't already exist.

The right argument **myda** is a vector of

monthnumber year [daynumbers] [usernumbers]

If no days are specified, all days are reported. If no user numbers are specified, the current user's number is assumed.

The result *R* is the tie number of the file to which signoff records, as requested in the right argument, have been appended in blocks of one hundred, or fewer, records.

For example:

10 *OFFHIST* 8 79
Append to file tied to 10 the requested records

'*OHRECS*' *OFFHIST* 1 79
Create a file called '*OHRECS*' and add records

'*OHRECS*' *OFFHIST* 2 79
Add more records to file '*OHRECS*'

If data is requested for a period for which there is no data on line, a single "pseudorecord" is returned, consisting of a single zero-filled record.

OHDEFN

Defines several variables which may be used to index the result of the function *OFFHIST* (see Appendix A).

R←*OHONLINE*

The result is an *N*×2 numeric matrix, where each row is a month-year pair indicating that *OFFHIST* data is on line and available for that month.

OHPRINT myda

The function *OHPRINT* formats into report form, and uses the function *OFFHIST* to format an annotated signoff history report. The right argument is a vector of:

monthnumber year [daynumbers] [user numbers]

If no days are specified, all days are reported. If no user numbers are specified, the current user's number is assumed.

For example:

OHPRINT 8 79
Print records for the accessor for August 1979

OHPRINT 6 79 25 26 27
Print records for 25, 26, and 27 June, 1979

OHPRINT 10 79 2381946 4511127
Print records for specified users for October 1979

OHPRINT 8 79 1 2 4511127 3918261
Print records for specified users on specified days in August 1979

R←file *OHPRINT* myda

The result *R* is the tie number of the file to which the signoff report has been appended. The left argument may be the number of a file which is already tied, or a file name which will be created if possible, and if it doesn't already exist.

For example:

10 *OHPRINT* 8 79
Append the *OFFHIST* report to the file tied to number 10

'*OHRECS*' *OHPRINT* 1 79
Create or tie the file *OHRECS* and append a report for January 1979

'*OHRECS*' *OHPRINT* 2 79
Append another report to the file *OHRECS*

R←*ONLINE*

The result is an *N*×2 numeric matrix where each row is a month-year pair indicating that usage data is on line and available for that month.

PRINT mya

Prints out a formatted report of the usage data as specified in the argument **mya**. *PRINT* calls the functions *BFDEFN*, *BF1DAYDEFN*, *NAMES*, *STORAGE*, *UPDATES*, and *USAGE* to obtain the necessary data.

The argument must be a vector of at least two elements, these elements denoting the month and year for which data is required.

- If the argument is just two elements, the report is for the accessor's usage data.

- If the argument is of more than two elements, the report is for those user numbers that follow the month and year.

Blank rows in the report indicate nonexistent or unavailable user numbers.

R←QUOTAS **a/f/w**

The result is a 9-element numeric vector containing information about the user's number. When the right argument *A* is empty (''), the result is:

R[⎕*IO*+0]	the user's internal flag: 1 if the user is 'internal'; 0 otherwise.
R[⎕*IO*+1]	the user's workspace quota: the maximum number of workspaces the user can save.
R[⎕*IO*+2]	the number of workspaces saved.
R[⎕*IO*+3]	the user's CPU limit. 0 indicates no limit.
R[⎕*IO*+4]	the user's File System volume class.
R[⎕*IO*+5]	the user's file quota: the maximum number of files the user can own.
R[⎕*IO*+6]	the number of files the user owns.
R[⎕*IO*+7]	the user's file reservation quota: the maximum number of bytes the user can reserve for files.
R[⎕*IO*+8]	the total number of bytes the user has reserved for files.

Workspace information alone may be obtained by using a right argument of **'w'**. In this case, *R*[⎕*IO*+ 4 5 6 7 8] will be zeros. Similarly, file information may be obtained by using **'f'** as the right argument, giving zeros in *R*[⎕*IO*+ 0 1 2 3]. *QUOTAS* executes more quickly if **'f'** or **'w'** is specified.

R←STORAGE **mya**

The argument **mya** indicates the month, year and user numbers for which the last days' storage data is required. The argument must be a vector of at least two elements, denoting the month and year for which data is required.

- If the argument is two elements, the result is a numeric vector of storage data for the accessor.

- If the argument is more than two elements, the result is a numeric matrix of storage data; one row for each user number following the month and year.

The values contained in the columns of the matrix, or elements of the vector, in the result may be indentified by displaying the function *BF1DAYDEFN*.

Nonexistent or unavailable numbers are indicated by zero-filled rows.

R←UPDATES **my**

The argument **my** must be a two-element vector indicating the month and year for which the status of update is required.

The result is a 5×3 numeric matrix. Column 1 indicates the updating status, column 2 the date of update in *MMDDYY* format, and column 3 the time of update in *HHMMSS* format. The status is defined as:

```
¯1 - unavailable.
 0 - update in progress.
 1 - valid.
```

The first row relates to **storage updates**, second to **sessions** (e.g. connect and CPU), third to *FILEPRINT*, fourth to *HSPRINT*s and fifth to **sorts**.

R←USAGE **mya**

The argument **mya** must be similar to the argument to *STORAGE*: a vector of two or more elements.

The result is of the same shape and form as the result of *STORAGE*. However, the data returned relates to cumulative (usually month-to-date) usage data, rather than last day's storage data.

The values contained in the result may be identified by displaying the function *BFDEFN*.

R←WHOIS **user**

The argument **user** is a scalar, vector or matrix, containing the user numbers or user names for which the user names, branch codes, and customer codes are required. Multiple user names in the argument may be in vector form separated by semicolons (;), or in matrix form where each row is a user name.

The result is an *N*×38 character matrix of names, one row for each user number or user name in the argument. The format of the result is like that of *□NAMES* plus two fields, each eight wide, of code information.

If the argument is numeric, any numbers that are not in the APL system or are unavailable to the accessor result in blank-filled corresponding rows.

Information for deleted users is not normally printed when the argument is a user name, but these records may be displayed by preceding the user name with a tilde (~).

For example:

WHOIS 1234567
Get information for user 1234567

WHOIS 1234567 7654321
Get information for user numbers 1234567 and 7654321

WHOIS 'ABC'
Get information for all users whose user names begin with *'ABC'*, ignoring deleted accounts

WHOIS 'ABCD;EFGH'
Get information for all users whose user names begin with either *'ABCD'* or *'EFGH'*, ignoring deleted accounts

WHOIS 'ABCD;~EFGH;IJK'
Get information for all users whose user names begin with *'ABC'*, ignoring deleted accounts; or *'EFGH'*, including deleted accounts; or *'IJK'*, ignoring deleted accounts

WHOIS 2 3 ⍴*'ABCDEFGHI'*
Get information for all users whose user names begin with *'ABC'*, *'DEF'*, or *'GHI'*, ignoring deleted accounts.

```
     ∇ BFDEFN;I
[1]   ⍝ COLUMN NUMBERS FOR THE MONTH-TO-DATE USAGE DATA
[2]   ⍝ CPU TIMES ARE IN 300THS OF UNITS
[3]   ⍝ CONNECT TIMES ARE IN 300THS OF A SECOND
[4]    I←⎕IO                    ⍝ USE CURRENT INDEX ORIGIN
[5]    BFUSER←I+0               ⍝ USER NUMBER
[6]   ⍝ COLUMNS 1 TO 6 ARE RESERVED (ORIGIN = 0)
[7]    BFCPU←I+7                ⍝ TERMINAL CPU TIME THIS MONTH
[8]    BFCON←I+8                ⍝ TERMINAL CONNECT TIME THIS MONTH
[9]    BFCHRS←I+9               ⍝ CHARACTERS THIS MONTH
[10]   BFNCPU←I+10              ⍝ N-TASK CPU TIME THIS MONTH
[11]   BFNCON←I+11              ⍝ N-TASK CONNECT TIME THIS MONTH
[12]   BFBCPU←I+12              ⍝ B-TASK CPU TIME THIS MONTH
[13]   BFBCON←I+13              ⍝ B-TASK CONNECT TIME THIS MONTH
[14]   BFBSSBD←I+14             ⍝ FILE BYTE DAYS THIS MONTH
                                  (÷1000 IFF 1=2|BFSTRUPD)
[15]   BFWSDYS←I+15             ⍝ WS DAYS THIS MONTH
[16]   BFWSBDYS←I+16            ⍝ WS BYTE DAYS THIS MONTH
[17]   BFSTRUPD←I+17            ⍝ STORAGE ACCOUNTING FLAGS
[18]   BFSOHUPD←I+18            ⍝ SOH UPDATE INDICATOR
[19]   BF30CPS←I+19             ⍝ TIME CONNECTED AT 30 CPS
[20]   BFLINES←I+20             ⍝ NUMBER OF LINES FILEPRINTED
                                  THIS MONTH
[21]   BFLNUPD←I+21             ⍝ FILEPRINT UPDATE INDICATOR
[22]   BFFORMS←I+22             ⍝ THE NUMBER OF FORMS MOUNTS THIS
                                  MONTH
[23]   BFTRAINS←I+23            ⍝ THE NUMBER OF TRAIN MOUNTS THIS
                                  MONTH
[24]   BFMISC←I+24              ⍝ MISCELLANEOUS FILEPRINT
                                  CHARGES THIS MONTH (CENTS)
[25]   BFLNSMIN←I+25            ⍝ THE NUMBER OF MINIMUM COST
                                  FILEPRINTS AND HSPRINTS
[26]   BFPRINTN←I+26            ⍝ THE NUMBER OF FILEPRINTS
                                  AND HSPRINTS
[27]   BFPLY←I+27               ⍝ THE NUMBER OF EXTRA PLY
                                  LINES (LINES=PLY) FOR
                                  HSPRINT AND FILEPRINT
[28]   BFHSUPD←I+28             ⍝ THE HSPRINT UPDATE
                                  INDICATOR
[29]   BFSRTUPD←I+29            ⍝ THE FILE SORT UPDATE
                                  INDICATOR
[30]   BFSRTN←I+30              ⍝ THE NUMBER OF FILE SORTS
                                  THIS MONTH
[31]   BFSRTMIN←I+31            ⍝ THE NUMBER OF MINIMUM COST FILE
                                  SORTS THIS MONTH
[32]   BFSRTOPS←I+32            ⍝ THE NUMBER OF FILEOPS
                                  DURING SORTING THIS MONTH
[33]   BFSRTSZE←I+33            ⍝ THE FILE SIZE TOTAL SORTED THIS
                                  MONTH
[34]   BFTPELNS←I+34            ⍝ THE NUMBER OF LINES PRINTED
```

```
                                    TO TAPE
[35]  BFDEC←I+35                  ⍝ THE NUMBER OF PRINTS
                                    DECOLLATED
[36]  BFFCOL←36                   ⍝ THE NUMBER OF COLUMNS IN
                                    MONTHLY DATA
[37]  BFFSHP←BFFL,BFFCOL          ⍝ THE SHAPE OF MONTHLY DATA
                                    BLOCKS
      ∇


      ∇ BF1DAYDEFN;I
[1]    ⍝COLUMN NUMBERS FOR THE LAST DAYS STORAGE DATA
[2]    I←⎕IO                      ⍝ USE CURRENT INDEX ORIGIN
[3]    BF1DUSER←I+0               ⍝ USER NUMBER
[4]    BF1DWSS←I+1                ⍝ NUMBER OF WORKSPACES SAVED
[5]    BF1DWSBYTES←I+2            ⍝ NUMBER OF BYTES OF
                                    WORKSPACES SAVED
[6]    BF1DBSSBYTES←I+3           ⍝ NUMBER OF FILE BYTES STORED
[7]    BF1DCOL←4                  ⍝ THE NUMBER OF COLUMNS IN
                                    THE 1 DAY BLOCKS
[8]    BF1DAYSHP←BF1DAYL,BF1DCOL  ⍝ THE SHAPE OF 1 DAY BLOCKS
      ∇


      ∇ OHDEFN;I
[1]    ⍝ COLUMN NUMBERS FOR THE OFFHIST DATA
[2]    ⍝ CPU TIMES ARE IN 300THS OF UNITS
[3]    ⍝ CONNECT TIMES ARE IN 300THS OF A SECOND
[4]    I←⎕IO                      ⍝ USE CURRENT INDEX ORIGIN
[5]    OHUSER←I+0                 ⍝ USER NUMBER
[6]    OHDATE←I+1                 ⍝ DATE IN FORM MMDDYY
[7]    OHTIME←I+2                 ⍝ TIME IN FORM HHMMSS
[8]    OHTASKID←I+3               ⍝ TASKID NUMBER
[9]    OHNODE←I+4                 ⍝ NODE/LINE NUMBER
[10]   OHTTYPE←I+5                ⍝ TASK TYPE. 0-T, 1-N, 2-B
[11]   OHBAUD←I+6                 ⍝ NOMINAL TERMINAL BAUD RATE
[12]   OHMODE←I+7                 ⍝ SIGN-OFF MODE. 1-BOUNCE,
                                    2-DROP, 3-CRASH, 4-CRASH,
                                    BUT NOT BILLED
[13]  ⍝ COLUMNS 8 AND 9 ARE RESERVED (ORIGIN = 0)
[14]   OHCPUS←I+10                ⍝ CPU DURING SESSION
[15]   OHCPUC←I+11                ⍝ CPU CUMULATIVE
[16]   OHCONS←I+12                ⍝ CONNECT TIME DURING SESSION
[17]   OHCONC←I+13                ⍝ CONNECT TIME CUMULATIVE
[18]   OHCHARS←I+14               ⍝ CHARACTER TRANSMISSION
                                    DURING SESSION
[19]   OHCHARC←I+15               ⍝ CHARACTER TRANSMISSION
                                    CUMULATIVE
[20]   OHWS←I+16                  ⍝ NUMBER OF WORKSPACES STORED
[21]   OHCR←I+17                  ⍝ NUMBER OF CARRIAGE RETURNS
                                    DURING SESSION
[22]   OHFOP←I+18                 ⍝ NUMBER OF FILE OPERATIONS
      ∇
```

A

ACCOUNTS function, description of 3-4
Access to usage data, with or without mask 3
ACCESS function, description of 3

B

BCCC function, description of 4
BFDEFN function
  description of 4
  definition of 11-12
BF1DAYDEFN function
  description of 4
  definition of 12
Branch code mask. See Mask
Branch code
  determining a user's 9
  information contained in 2
Byte reservation, determining a user's 8

C

Codes associated with user numbers
  by mask 4
  by user number 4
Code
  branch, overview of 2
  customer, overview of 2
  indication of range of. See Mask
  information contained in branch and customer 2
  zero-filled 2
Cpu limit, determining a user's 8
CUSTADDR function, description of 4
Customer address, determining 4
Customer code mask. See Mask
Customer code, determining a user's 9
Customer number, of customer code 2

D

Determining code groups 4-5

F

Fields in branch/customer codes, requirements of 2
File quota, determining a user's 8
File reservation quota, determining a user's 8
Files, determining number owned by a user 8

G

GROUPS function, description of 4-5

I

Indexing the result of
  OFFHIST 6
  STORAGE 4
  USAGE 4
Internal flag, determining a user's 8
Invoice subgroup, definition of 2

M

Mask
  access with or without 3
  combining branch and customer code 3
  examples of branch code 2
  examples of customer code 3
  use of 2

N

NAMES function, description of 5
Names associated with account numbers, determining 5

O

OFFHIST data, determining availability of on-line 6
OFFHIST function
  indexing the result of (*OHDEFN*) 6
  desciption of 5-6
OHDEFN function
  definition of 12
  description of 6
OHONLINE function, description of 6
OHPRINT function, description of 6-7
OHLINE function, description of 7

P

PRINT function, description of 7-8

Q

QUOTAS function, description of 8

R

Reports
  generating formatted sign-off history 6
  generating formatted usage data 7

S

Saved workspaces, determining number of user's 8
Sign-off history report
  appending to file 7
  generating a formatted 6
Sign-off records
  appending to file 6
  reports of 5
Storage data
  of accessor, determining the 8
  of users, determining the 8
*STORAGE*function
  description of 8
  indexing the result of (*BF1DAYDEFN*) 4

U

UPDATES function, description of 9
Update status, determining the 9
Usage data
  access with and without mask 3
  default access to 3
  determining availability of 7
  determining cumulative 9
  generating a formatted report of 7
USAGE function
  description of 9
  indexing the result of (*BFDEFN*) 4
User access control mask, determining 3

V

Volume Class, determining a user's 8

W

WHOIS function, description of 9-10
Workspace quota, determining a user's 8